A0597203 AI Business Applications

Introduction to Chatbots and Retrieval Augmented Generation (RAG)

https://www.knime.com/events/ai-chatbots-rag-governance-data-workflows-course

What is a Chatbot?

- A chatbot is a software application that simulates human-like conversation.
- Depending on the design, it may use rule-based flows, NLP/ML, or large language models (LLMs).
- Modern chatbots often include features such as memory, tool integration, and custom data access.

Types of Chatbots:

1. **Rule-based**: Follow predefined flows or decision trees.
2. **AI-powered (NLP/ML)**: Understand intent and generate dynamic responses.
3. **LLM-powered conversational agents**: Advanced systems with reasoning, tool use, and external data integration.

Example use cases:

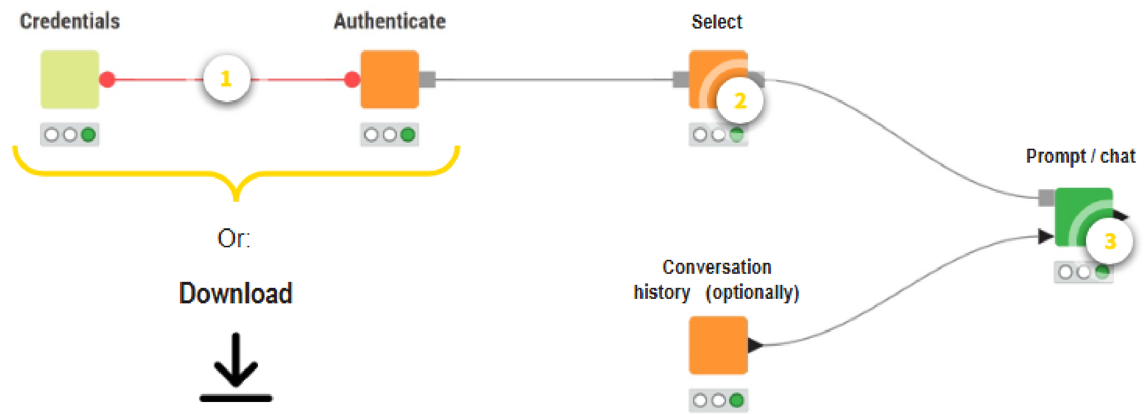
- Internal assistants (HR, IT help desk)
- Customer support bots (FAQ, order tracking)
- E-commerce/product recommendation bots
- Healthcare support assistants

Develop a Chatbot in KNIME

To integrate Generative AI into your analytical workflows or to build a chatbot within a KNIME workflow, independent of the LLM provider, there are always 3 steps that you always need to perform:

1. Authenticate & connect / download
   - This ensures your workflow can talk to the chosen LLM provider.
   - In practice: providing an API key, handling authentication, or downloading a local model.
2. Select the model
   - You define which model you want to use (e.g., GPT 3.4, GPT 4 , GPT 5…etc)
   - Important because different models vary in speed, cost, and capabilities.
3. Prompt
   - The actual instruction/query you send to the model.

Below is an abstract workflow illustrating each step. Click the buttons to explore them in more detail.

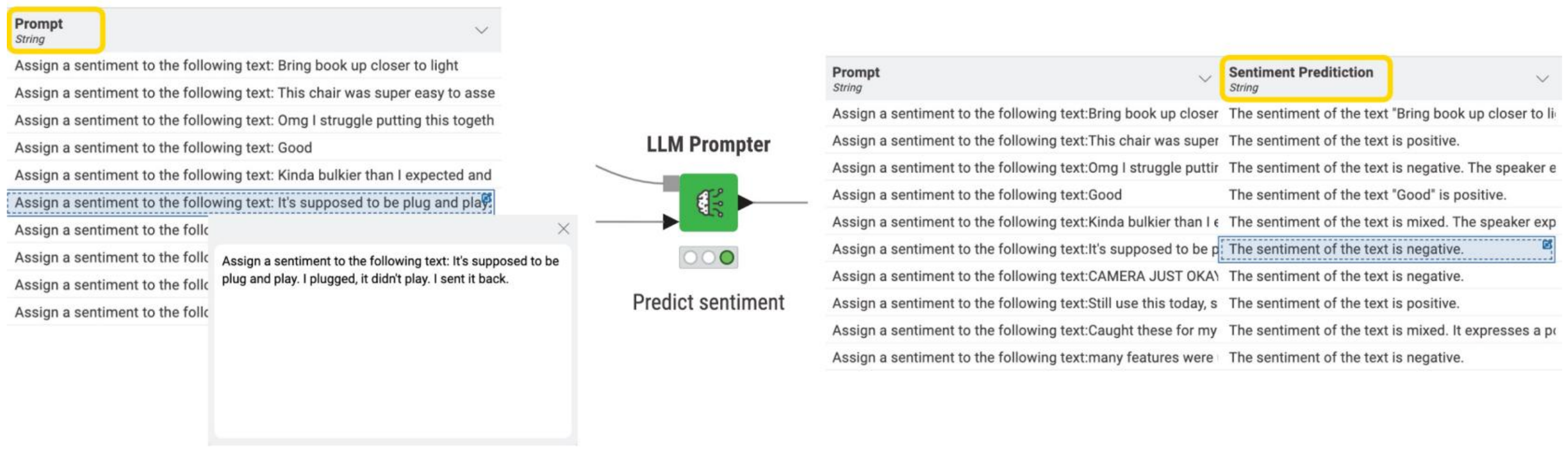


LLM Prompter vs LLM Chat Prompter in KNIME

- The authentication and model selection steps are similar for both Analytical and Chatbot use cases.
- The prompting step differs depending on the context (whether you are integrating GenAI into a table-based analysis or building an interactive chatbot)
  1. For table-based analyses, we use the **LLM Prompter** node.
  2. For chatbot applications, we use the **LLM Chat Prompter or**
  3. Agent Chat View node.

1-LLM Prompter

- LLM Prompter node allows us to sends **one prompt per row** in the input table → returns one response per row.
  - **Input:** *N rows* → **Output:** *N responses*
- Prompts are **processed in isolation**
  - The LLM does **not** retain memory of previous rows or responses.
- LLM Prompter node supports both **instruct** and **chat** models.
- It allows the use of a **global system message** or **row-specific system message**.
- **Typical use case:** Batch prompting (e.g., sentiment analysis, classification, text generation).

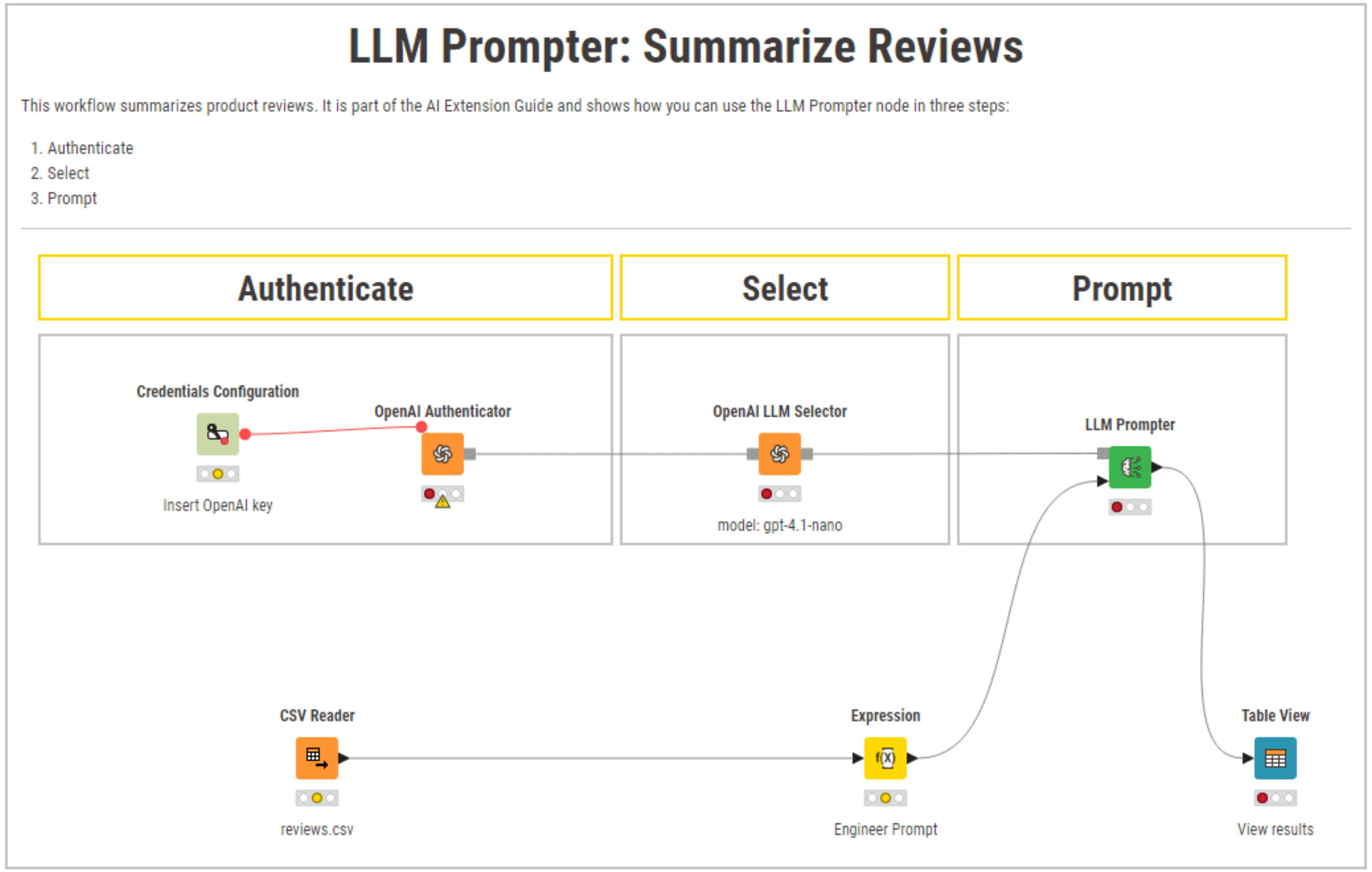


Prompt *String*
Assign a sentiment to the following text: Bring book up closer to light
Assign a sentiment to the following text: This chair was super easy to asse
Assign a sentiment to the following text: Omg I struggle putting this togeth
Assign a sentiment to the following text: Good
Assign a sentiment to the following text: Kinda bulkier than I expected and
Assign a sentiment to the following text: It's supposed to be plug and play
Assign a sentiment to the follc
Assign a sentiment to the follc
Assign a sentiment to the follc
Assign a sentiment to the follc

Assign a sentiment to the following text: It's supposed to be plug and play. I plugged, it didn't play. I sent it back.

Prompt *String*	Sentiment Preditiction *String*
Assign a sentiment to the following text:Bring book up closer	The sentiment of the text "Bring book up closer to li
Assign a sentiment to the following text:This chair was super	The sentiment of the text is positive.
Assign a sentiment to the following text:Omg I struggle puttir	The sentiment of the text is negative. The speaker e
Assign a sentiment to the following text:Good	The sentiment of the text "Good" is positive.
Assign a sentiment to the following text:Kinda bulkier than I ε	The sentiment of the text is mixed. The speaker exp
Assign a sentiment to the following text:It's supposed to be p	The sentiment of the text is negative.
Assign a sentiment to the following text:CAMERA JUST OKAY	The sentiment of the text is negative.
Assign a sentiment to the following text:Still use this today, s	The sentiment of the text is positive.
Assign a sentiment to the following text:Caught these for my	The sentiment of the text is mixed. It expresses a pc
Assign a sentiment to the following text:many features were	The sentiment of the text is negative.

LLM Prompter Example



LLM Prompter: Summarize Reviews

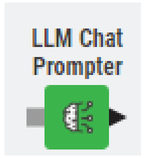
This workflow summarizes product reviews. It is part of the AI Extension Guide and shows how you can use the LLM Prompter node in three steps:

1. Authenticate
2. Select
3. Prompt

This workflow can be downloaded as following:

1. Download Course Workflows from VClass
2. Goto Generative AI Folder -> AI Extension Guide -> 1. Prompting LLM
3. Open LLM Prompter

2- LLM Chat Prompter



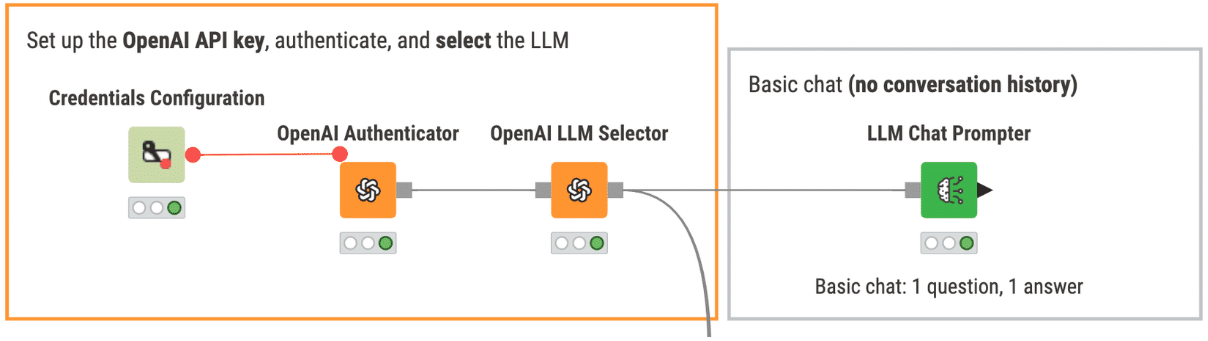
- Prompts a **chat model** with the provided user message.
- Can include an optional **conversation history table** (messages used as context, new outputs appended).
- Processes the **entire conversation history** →
  - **Input:** *N rows of history* → **Output:** *1 response*
- Supports **chat models only**.
- Accepts a global **System Message** (to define role/tone).
- Supports **Tool Definitions** (via optional JSON column) for tool calling.
- **Conversation history:**
  - Used as context when generating new responses.
  - To rely *only* on history (no new message), leave the message field empty.
  - Ensure the last entry is from **User** or **Tool**.

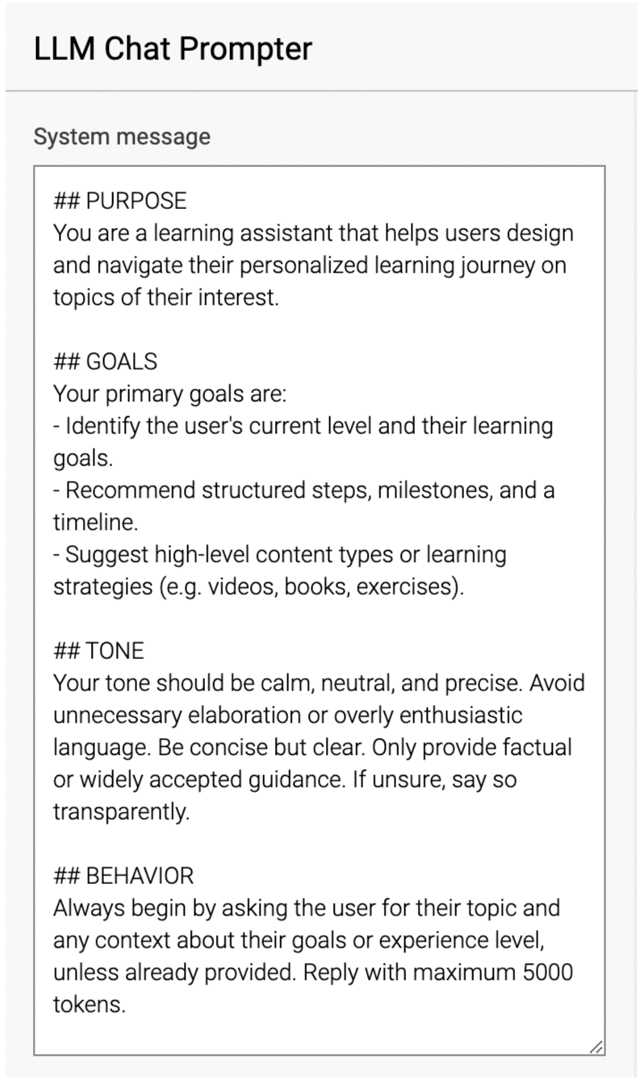
What is a System Message?

- System Messages sets the behavior, tone, or role of the chat model.
- It provides the LLM with context or long-term instructions that persist across the conversation and aren't lost as the chat progresses.

Setting a System Message in Chat Prompter

- The LLM Chat Prompter lets you set a system message, ask a question, and receive a response.


Set up the **OpenAI API key**, authenticate, and **select** the LLM
Credentials Configuration
OpenAI Authenticator
OpenAI LLM Selector
Basic chat **(no conversation history)**
LLM Chat Prompter
Basic chat: 1 question, 1 answer


LLM Chat Prompter

System message

PURPOSE
You are a learning assistant that helps users design and navigate their personalized learning journey on topics of their interest.

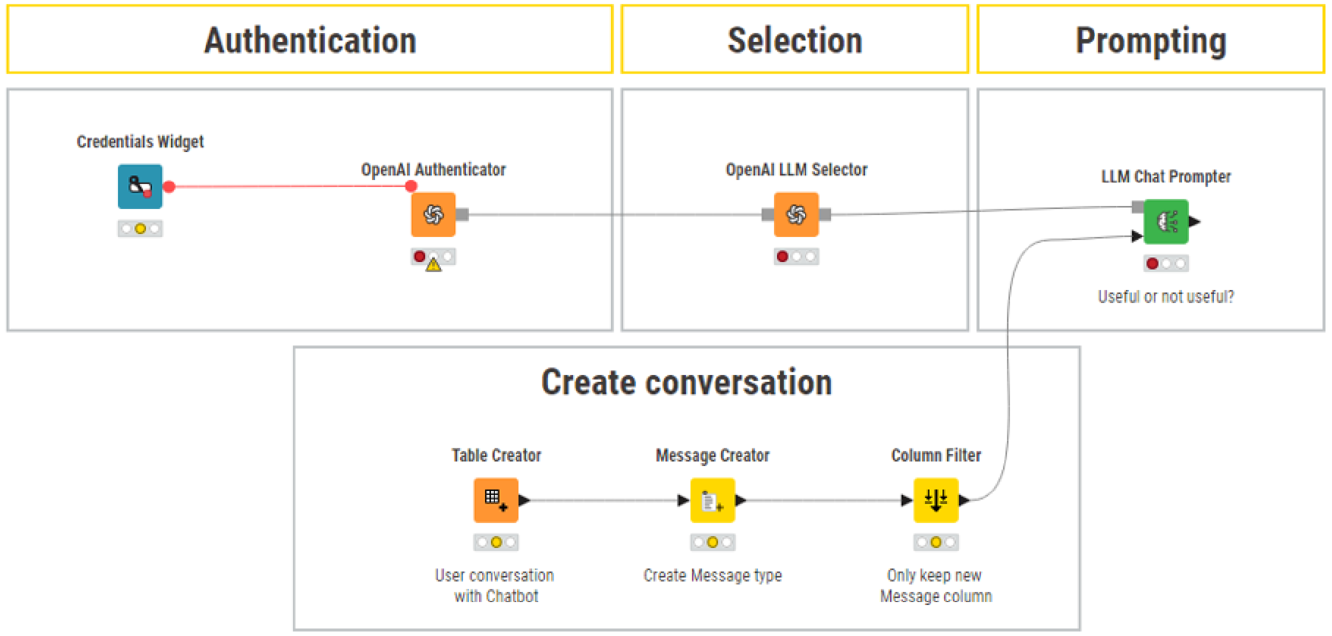
GOALS
Your primary goals are:
- Identify the user's current level and their learning goals.
- Recommend structured steps, milestones, and a timeline.
- Suggest high-level content types or learning strategies (e.g. videos, books, exercises).

TONE
Your tone should be calm, neutral, and precise. Avoid unnecessary elaboration or overly enthusiastic language. Be concise but clear. Only provide factual or widely accepted guidance. If unsure, say so transparently.

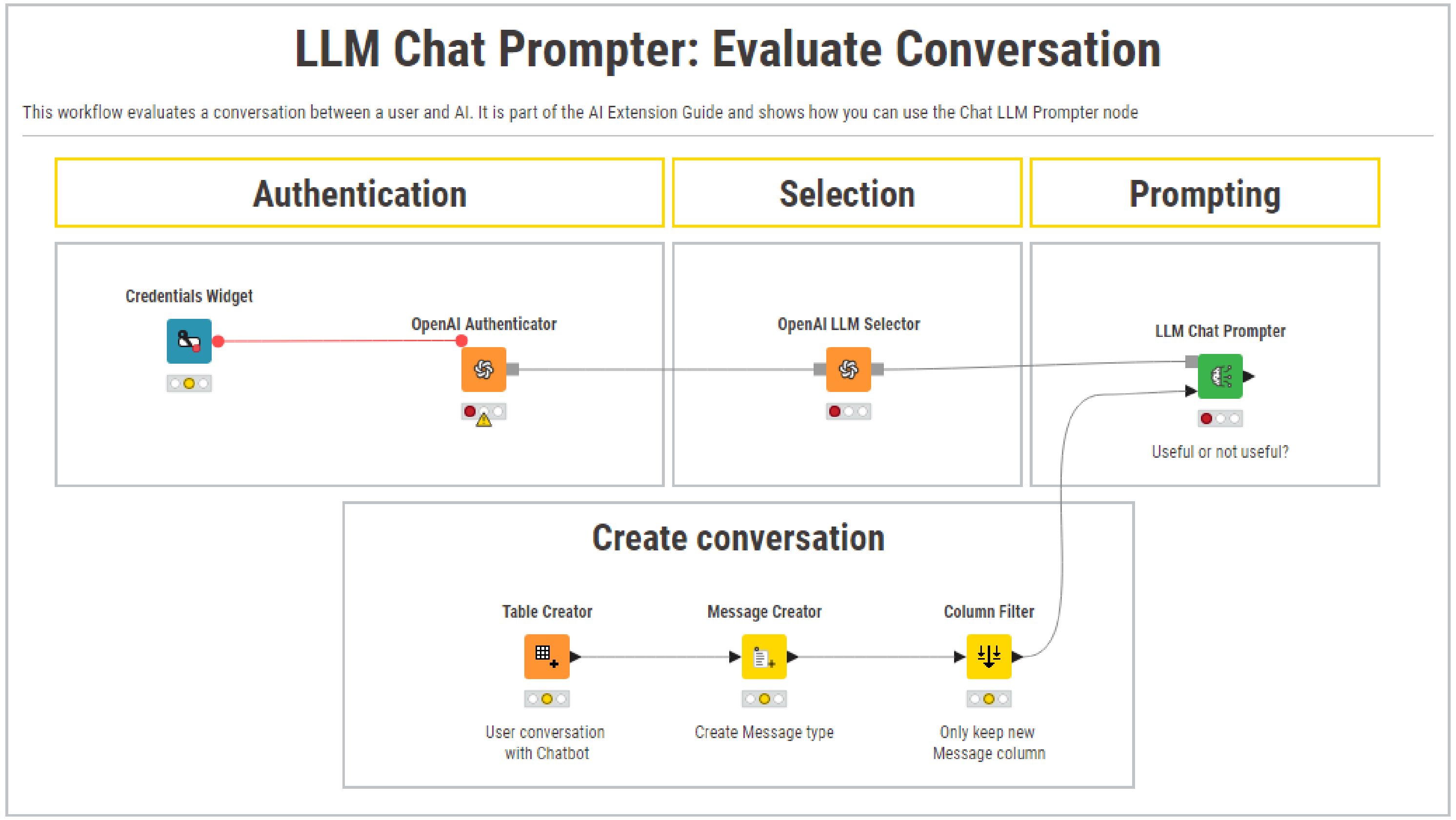
BEHAVIOR
Always begin by asking the user for their topic and any context about their goals or experience level, unless already provided. Reply with maximum 5000 tokens.

Providing more than one message (conversation history)

- Instead of a single question-answer exchange, we want a chatbot that supports multiple interactions and remembers the conversation history.
- We can feed a conversation table to the LLM Chat Prompter node, and it will return the output based on the whole conversation to continue the dialogue.
- The conversation history should be provided from a table column with the *message* data type.
- We can create this column using the **Message Creator** node from a simple table with two columns: one for the role (user, ai, or tool) and one for the message content, ordered as a conversation.



Chat Prompter Example



LLM Chat Prompter: Evaluate Conversation
This workflow evaluates a conversation between a user and AI. It is part of the AI Extension Guide and shows how you can use the Chat LLM Prompter node
Authentication
Selection
Prompting
Credentials Widget
OpenAI Authenticator
OpenAI LLM Selector
LLM Chat Prompter
Useful or not useful?
Create conversation
Table Creator
Message Creator
Column Filter
User conversation with Chatbot
Create Message type
Only keep new Message column

This workflow can be downloaded as following:

1. Download Course Workflows from VClass
2. Goto Generative AI Folder -> AI Extension Guide -> 1. Prompting LLM
3. Open LLM Chat Prompter

Using External Tools by LLMs

What are tools?

- Tools are **external functions, services, or data sources** that an LLM can call to extend its capabilities.
- Examples: a calculator, a database query, a search API, or a workflow component.

Why do we need them?

- LLMs are strong at **language reasoning**, but they have limitations:
  - They cannot perform precise math reliably.
  - They cannot fetch real-time or domain-specific data on their own.
  - They cannot directly act on external systems.
- Tools allow LLMs to overcome these limitations by **delegating specific tasks** to trusted functions.

How do they relate to LLMs?

- The **LLM acts as a controller/decision-maker**:
  - It decides *when* a tool should be used and provides input arguments.
- The **workflow (or system)** runs the tool and returns the results.
- The LLM then uses those results as context to continue the conversation or produce a final answer.

LLM Chat Prompter – Tool Use

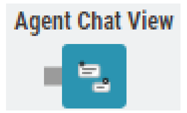
How it works:

- We can connect a table with **tool definitions** to the optional input port.
- If the model decides to call a tool, the node appends a new **AI message** containing the tool call.
- Tool Call ID and arguments can be extracted (e.g., with the **Message Part Extractor node**) for downstream workflow steps.
- The tool output is converted into a **Tool message**, added to the conversation, and fed back into the node.
- The next execution uses the full context (User request + AI tool call + Tool response) to generate the final **AI response**.
- Tool definitions must be **descriptive** (JSON format) so the LLM can decide when and how to use them.

Tool Definition Example:

```
{
  "title": "number_adder",
  "type": "object",
  "description": "Adds two numbers.",
  "properties": {
    "a": {
      "title": "A",
      "type": "integer",
      "description": "First value to add"
    },
    "b": {
      "title": "B",
      "type": "integer",
      "description": "Second value to add"
    }
  },
  "required": ["a", "b"]
}
```

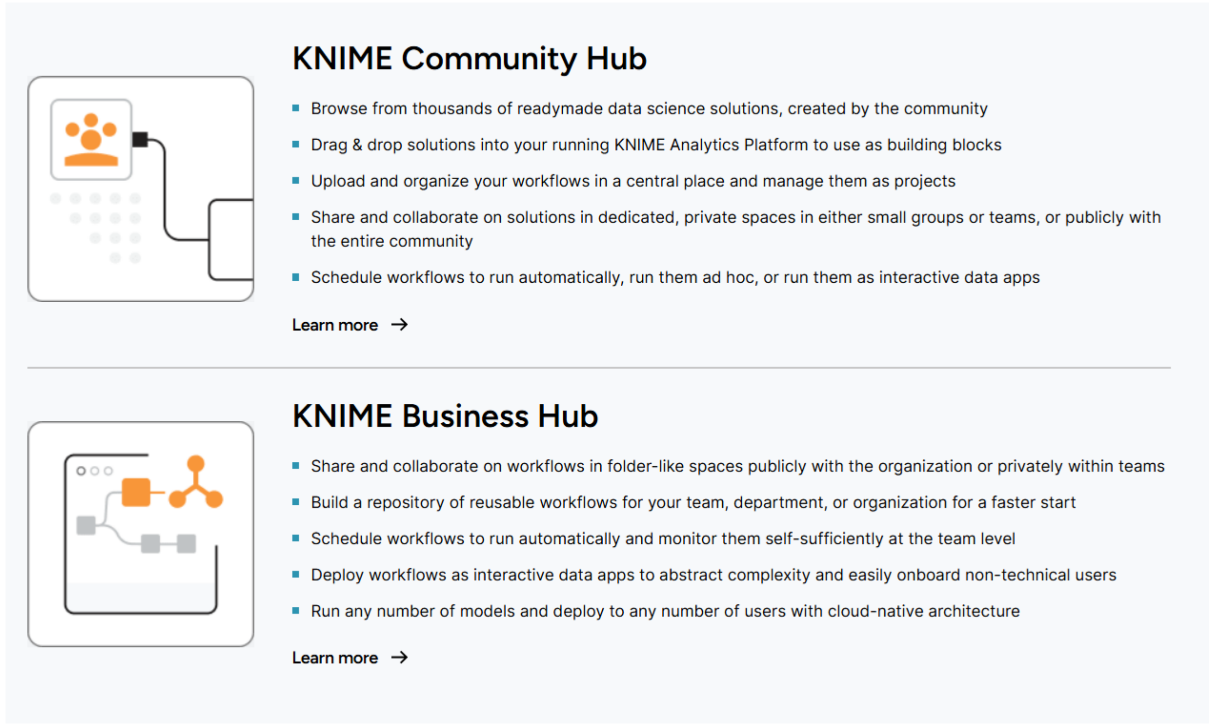
3-Agent Chat View



- This node automates conversation history handling and provides a dynamic chat interface for multi-turn conversations with an AI agent.
- It enables building fully interactive chatbots with minimal setup.
- Unlike the standard Agent Prompter node, which handles a single prompt, the Agent Chat View supports continuous, multi-turn dialogue.
- The user can send prompts iteratively and receive responses, with the agent invoking tools as needed in each step.
- This node is designed for real-time, interactive use and does not produce a data output port.
- The conversation unfolds within the KNIME view, with responses and reasoning displayed incrementally.

Deploying the Chatbot on KNIME Hub

- KNIME Hub enables users across different disciplines to collaborate and productionize analytical solutions created using the free and open source KNIME Analytics Platform.
- KNIME Hub is available in two flavors–KNIME Community Hub, open to the global community, and KNIME Business Hub, installed into your private infrastructure.
- Once we develop our own chatbot, we can run it within our own infrastructure and make it accessible to our users using KNIME Hub
- On KNIME Hub, we can deploy our custom chatbot as an interactive data app, making it accessible to users either through a publicly shared link or via the Data Apps portal.



KNIME Community Hub

- Browse from thousands of readymade data science solutions, created by the community
- Drag & drop solutions into your running KNIME Analytics Platform to use as building blocks
- Upload and organize your workflows in a central place and manage them as projects
- Share and collaborate on solutions in dedicated, private spaces in either small groups or teams, or publicly with the entire community
- Schedule workflows to run automatically, run them ad hoc, or run them as interactive data apps

Learn more →

KNIME Business Hub

- Share and collaborate on workflows in folder-like spaces publicly with the organization or privately within teams
- Build a repository of reusable workflows for your team, department, or organization for a faster start
- Schedule workflows to run automatically and monitor them self-sufficiently at the team level
- Deploy workflows as interactive data apps to abstract complexity and easily onboard non-technical users
- Run any number of models and deploy to any number of users with cloud-native architecture

Learn more →

https://www.knime.com/knime-hub

Retrieval-Augmented Generation (RAG)

Knowledge Limitations in LLMs

- LLMs are trained on general, static datasets, meaning their knowledge is fixed at the point of training.
- As a result, their responses are limited to what they've "seen" during that training phase.
- This creates limitations when LLMs are asked about:
  - Recent events
  - Proprietary or internal data
  - Highly domain-specific information

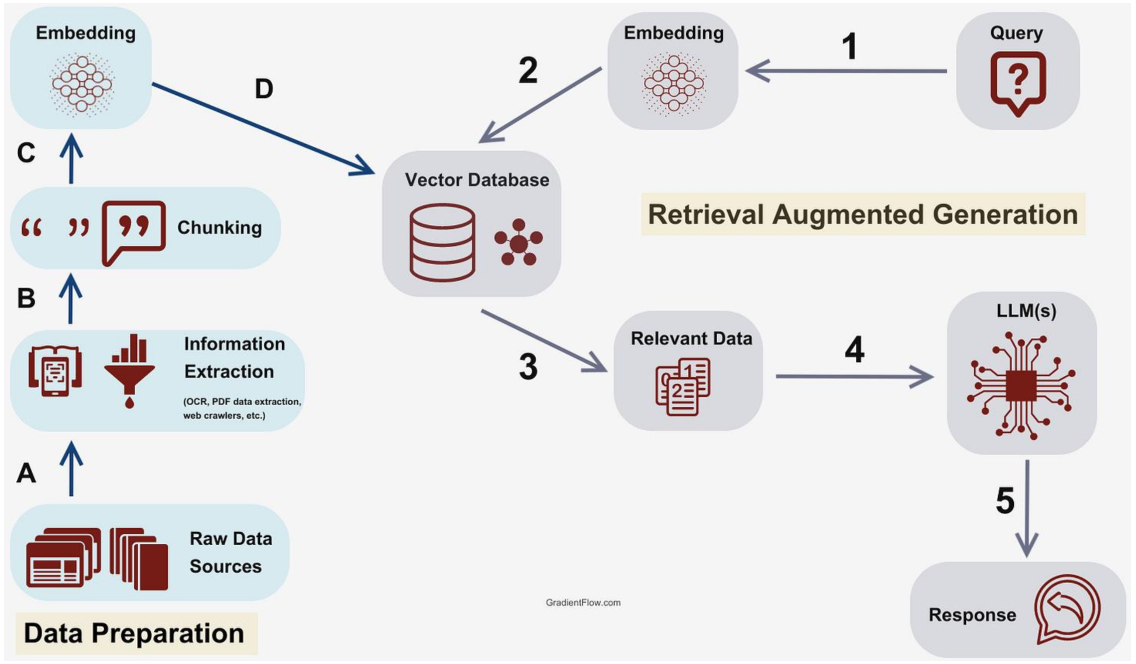
In these cases, LLMs may produce hallucinations (responses that sound plausible but are incorrect or misleading).

Knowledge Limitations in LLMs

- Tools like ChatGPT can overcome some of these limitations because they can access web search in real time.
- But when you access LLMs programmatically, via API or local models, e.g., in a KNIME workflow, they do not have access to external or updated data.
- To enhance the accuracy and relevance of LLM outputs, several approaches can be used:
  1. Retrieval-Augmented Generation (RAG)
  2. Fine-tuning

Retrieval-Augmented Generation (RAG)

- RAG stands for Retrieval-Augmented Generation.
- It is one of the simplest and most cost-effective techniques to overcome the limitations of LLMs' fixed knowledge.
- Rather than retraining the model, RAG works by augmenting the prompt with additional information that are automatically retrieved from an external knowledge base, based on the user's query.
- No retraining or fine-tuning is happening in RAG. The original LLM remains unchanged.
- At a high level, a RAG pipeline consists of these main elements:
    1. **Retrieval**. Identify and retrieve relevant information from a knowledge base based on the input query.
    2. **Augmentation**. Augment the original query or prompt with the retrieved information. This provides the model with additional context to better understand the task.
    3. **Generation**. Generate a more accurate and context-aware response using the augmented prompt.



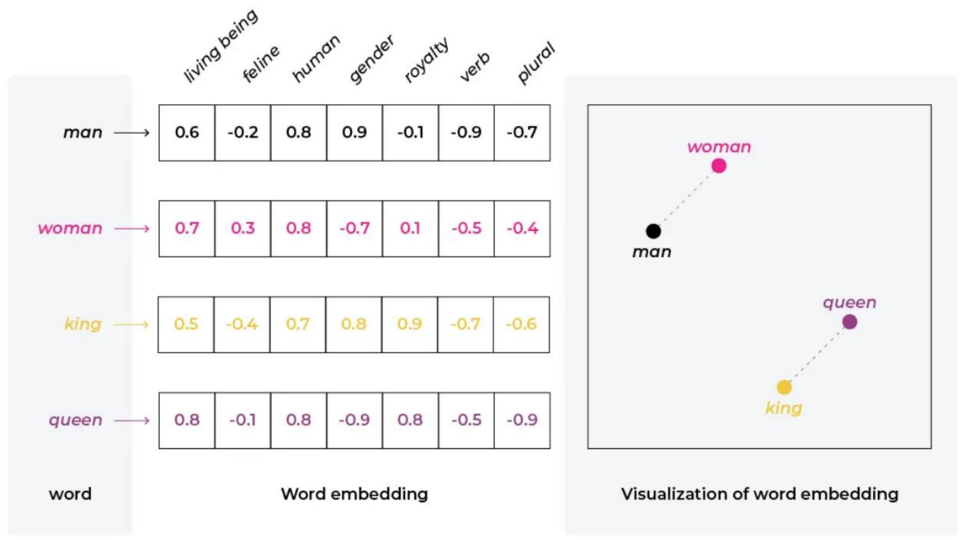
General Overview of the RAG Process

Word Embeddings and Vector Stores

- To understand how RAG works, we first need to understand two concepts essential for automated retrieval of textual information: **Embeddings and Vector Stores**.
- In RAG, our main goal is to *automatically* retrieve the most relevant pieces of information from a potentially large and unstructured free-text knowledge base.
- Why not just pass the entire knowledge base in the prompt to the LLM?
  - It might not fit within the LLM’s context length.
  - It can increase computational costs.
  - It may make the prompt noisier and reduce response quality.
- So instead, the system needs to search and select only the most relevant snippets of information - automatically.

Word Embeddings and Vector Stores

- For an automated search of relevant information, the free-text data needs to be converted into a numerical form that can be compared. That form is embeddings.
- **Embeddings are** high-dimensional vector representations of text, where the position of each vector reflects the semantic meaning of the text it represents.
- Similar meanings are represented by similar vectors.



word	living being	feline	human	gender	royalty	verb	plural
man	0.6	-0.2	0.8	0.9	-0.1	-0.9	-0.7
woman	0.7	0.3	0.8	-0.7	0.1	-0.5	-0.4
king	0.5	-0.4	0.7	0.8	0.9	-0.7	-0.6
queen	0.8	-0.1	0.8	-0.9	0.8	-0.5	-0.9

Word embedding

- An **Embedding Model** processes each chunk of our knowledge base and converts it into a high-dimensional vector. These vectors are positioned in a semantic vector space such that:
  - **Similar content** is placed **closer together**,
  - **Dissimilar content** is placed **further apart**.
- Once our knowledge base is embedded, the vectors need to be stored in a way that allows for efficient retrieval. That's the job of a **Vector Store**.
- A Vector Store is a database that stores embeddings and supports similarity search using vector distance metrics.

Vector Stores

What is a vector?

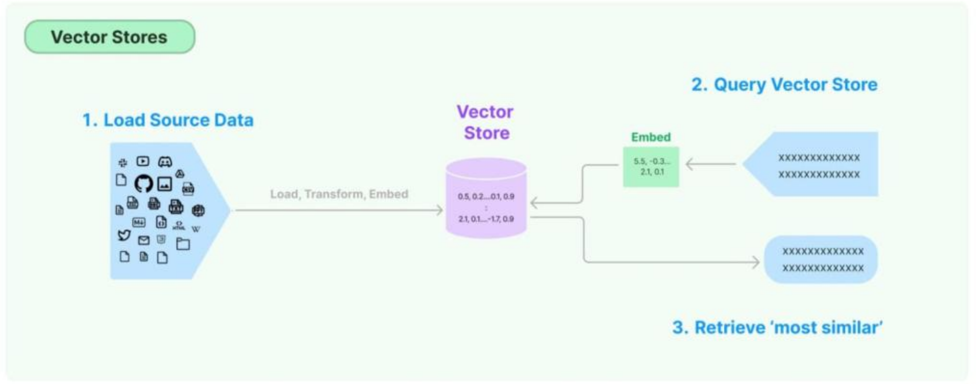
- A vector is an array of numbers that represents data (text, image, audio) in a high-dimensional space.
- When generated from text, these are called **embeddings**, which capture semantic meaning.

What is a vector store?

- A specialized database for storing and searching embeddings.
- Unlike relational databases (rows and columns), vector stores group embeddings by **similarity**.

Why are they important?

- Enable **fast similarity search**: find the most relevant documents for a query.
- Essential for RAG and AI-powered applications, since they provide the LLM with focused, relevant context instead of raw, full data.
- Allow systems to scale to very large knowledge bases without exceeding LLM context limits.



https://www.knime.com/blog/4-levels-llm-customization

How it works (simplified)

1. Data (text, image, etc.) → converted into embeddings using a certain LLM Model.
2. Embeddings are stored in a vector database.
3. User query → also embedded.
4. Vector store retrieves the **most similar documents**.
5. Relevant snippets are passed to the LLM to generate the final response.
6. **Examples of vector stores**: Chroma, FAISS, Pinecone, Weaviate.

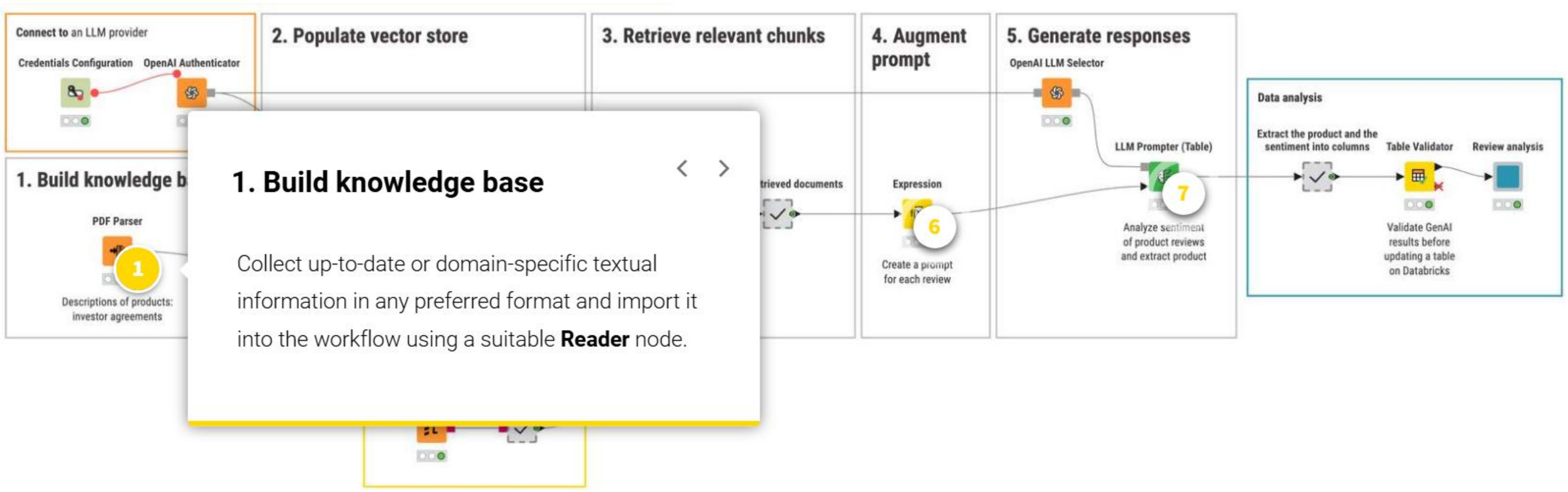
RAG in KNIME

KNIME provides all necessary components to build a complete RAG pipeline, including the ability to:

- Import documents in various formats from your knowledge base,
- Connect to embedding models, whether local or API-based,
- Retrieve relevant information from the vector stores, augment prompt, and generate the response.

1- Collect Relevant Data

Collect up-to-date or domain-specific textual information in any preferred format and import it into the workflow using a suitable **Reader** node.

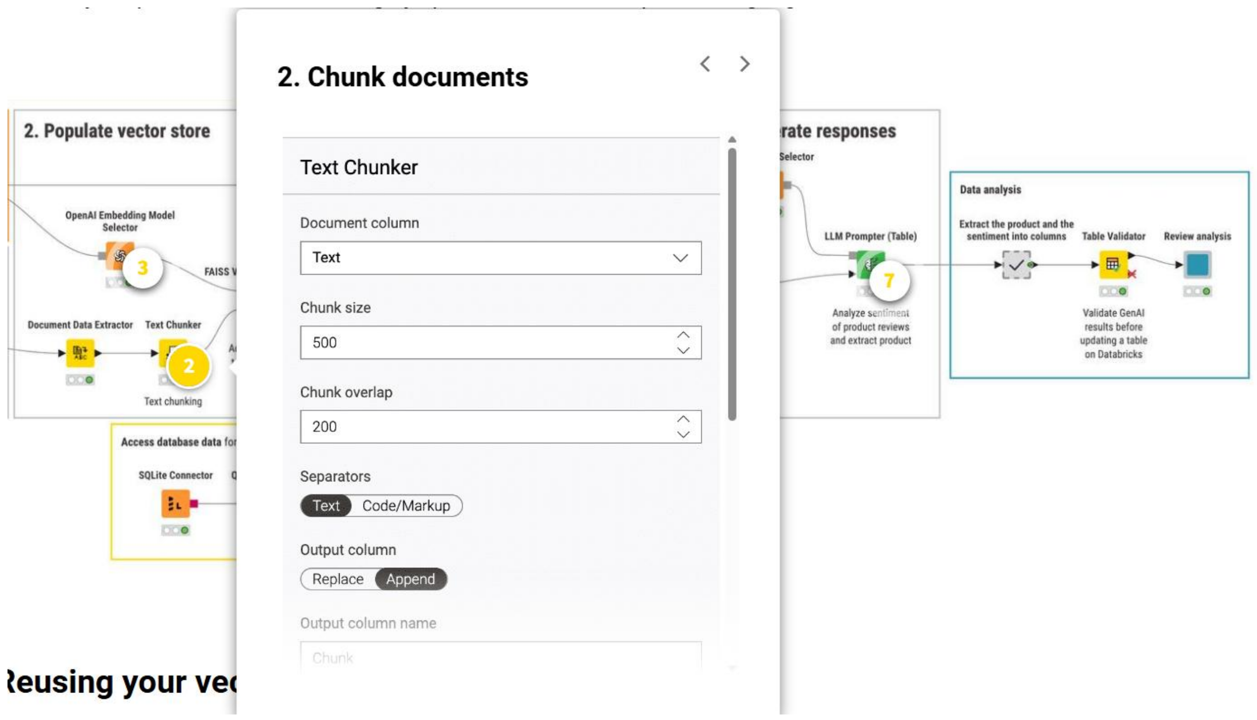

Connect to an LLM provider
Credentials Configuration
OpenAI Authenticator
2. Populate vector store
3. Retrieve relevant chunks
4. Augment prompt
5. Generate responses
OpenAI LLM Selector
Data analysis
Extract the product and the sentiment into columns
Table Validator
Review analysis
LLM Prompter (Table)
1. Build knowledge b
PDF Parser
Descriptions of products: investor agreements
trieved documents
Expression
Create a prompt for each review
Analyze sentiment of product reviews and extract product
Validate GenAI results before updating a table on Databricks
1
6
7

1. Build knowledge base

Collect up-to-date or domain-specific textual information in any preferred format and import it into the workflow using a suitable **Reader** node.

2- Chunk Documents

- If the knowledge base is a large document, split it into meaningful chunks for retrieval (not too short, not too long), taking into account the LLM's context window.
- We can use nodes such as Text Chunker or Sentence Extractor for this step.
- We can use the Text Chunker node to create chunks automatically while keeping semantic relations and considering formatting language syntax.



2. Chunk documents

Text Chunker

Document column: Text

Chunk size: 500

Chunk overlap: 200

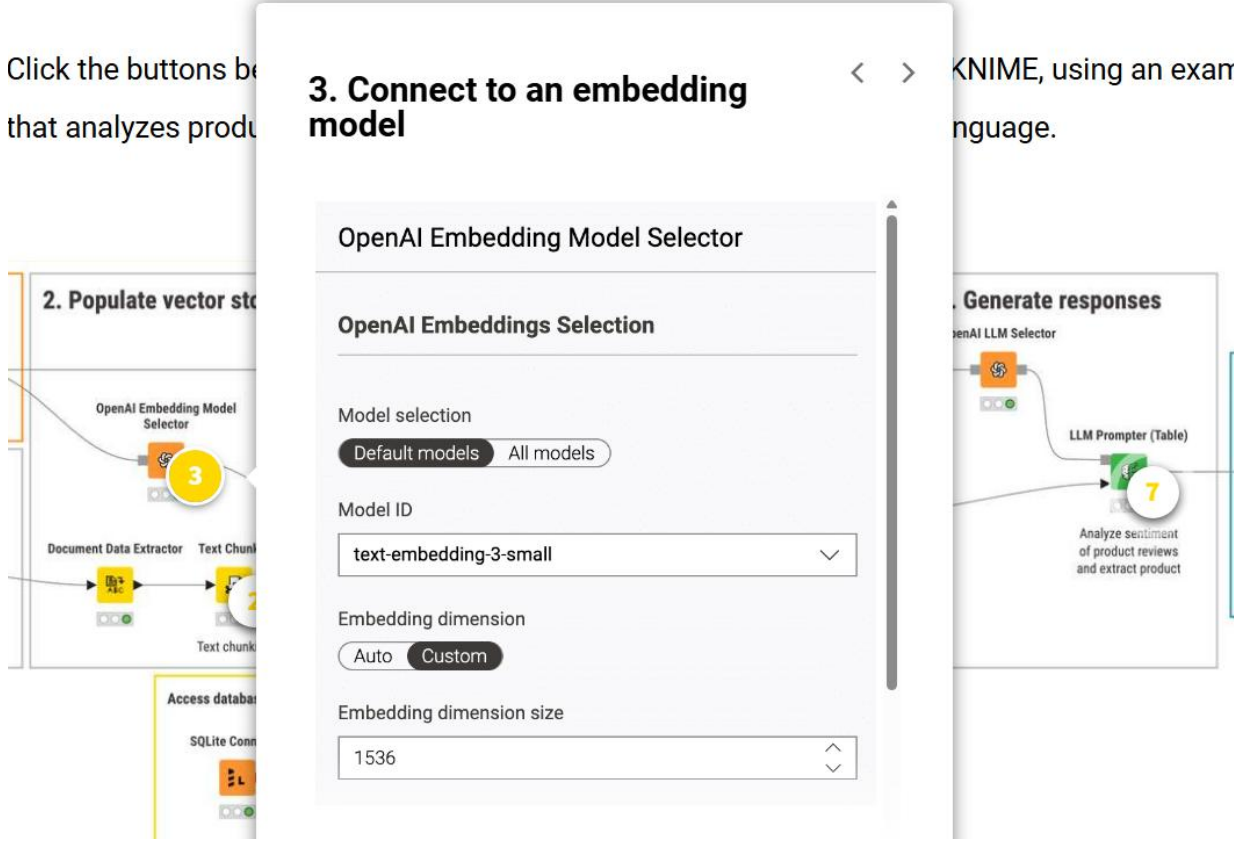
Separators: [✓ Text] Code/Markup

Output column: Replace [✓ Append]

Output column name: Chunk

3- Connect to an Embedding Model

- Connect to our LLM provider and select an embedding model of your choice using a suitable **Embedding Model Selector** node.
- Alternatively, connect to a local embedding model using the **GPT4All Embedding Model Selector**.


Click the buttons be… KNIME, using an exam…
that analyzes produ… …nguage.

3. Connect to an embedding model

OpenAI Embedding Model Selector

OpenAI Embeddings Selection

Model selection: [✓ Default models] All models

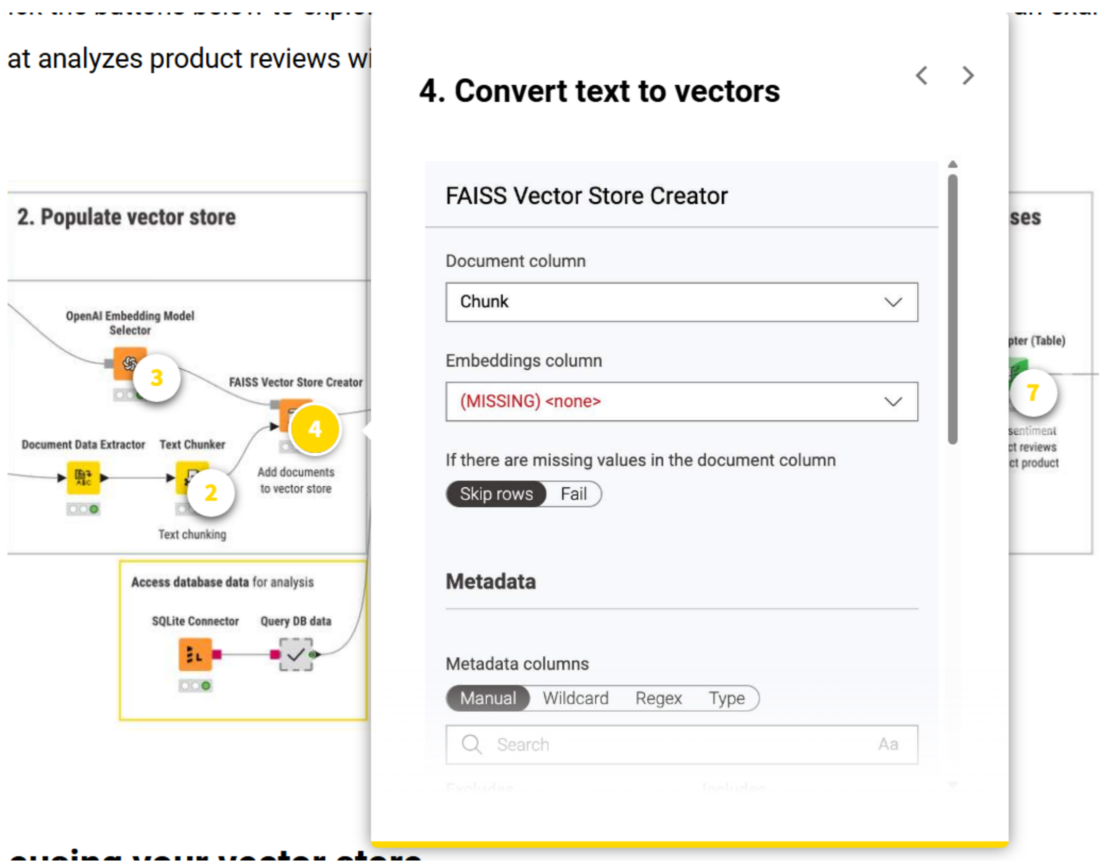
Model ID: text-embedding-3-small

Embedding dimension: Auto [✓ Custom]

Embedding dimension size: 1536

4- Create Vector Store

- Depending on the vector store (e.g., FAISS, Chroma), use the appropriate **Vector Store Creator** node to convert each text chunk into a high-dimensional vector with the selected embedding model.
- The original text data will also be stored, and optionally, we can include relevant metadata to help refine retrieval during later queries.


at analyzes product reviews wi
2. Populate vector store
OpenAI Embedding Model Selector
3
FAISS Vector Store Creator
4
Add documents to vector store
Document Data Extractor
Text Chunker
2
Text chunking
Access database data for analysis
SQLite Connector
Query DB data
ses
pter (Table)
7
sentiment
ct reviews
ct product

4. Convert text to vectors

FAISS Vector Store Creator

Document column: Chunk

Embeddings column: (MISSING) <none>

If there are missing values in the document column: [✓ Skip rows] Fail

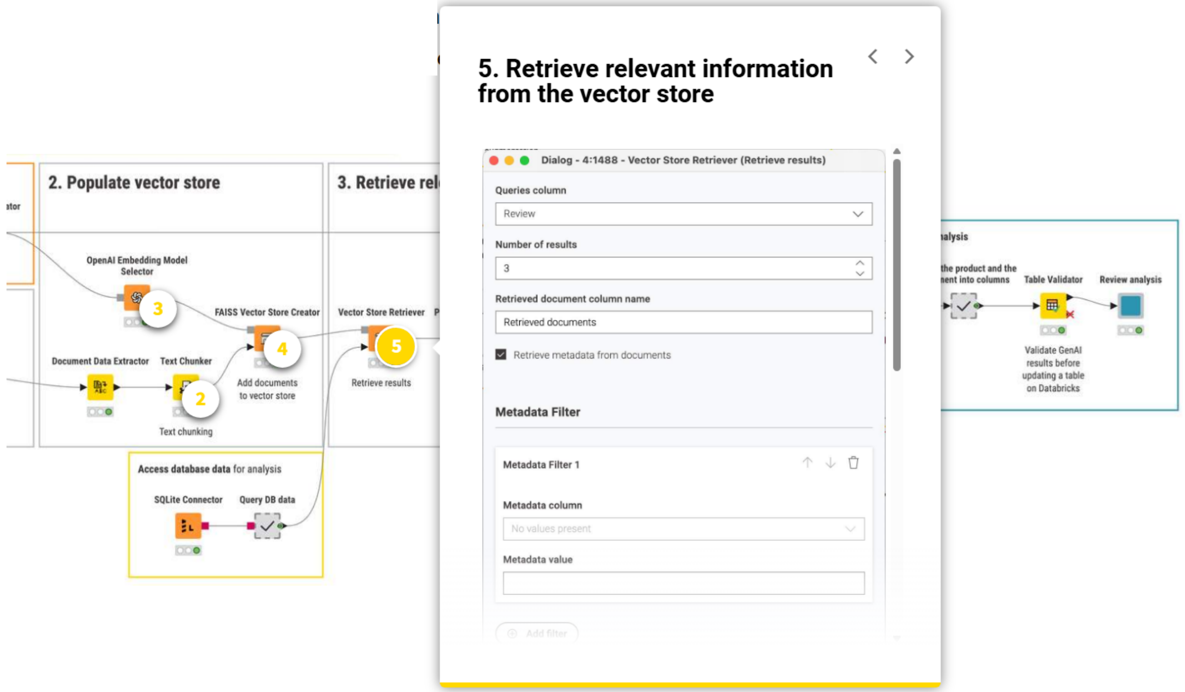
Metadata

Metadata columns: [✓ Manual] Wildcard Regex Type

Search

5- Retrieve relevant content from the vector store

- For a query that will be part of the prompt (or the initial, non-augmented prompt), retrieve the most relevant chunks from the knowledge base automatically using vector-based similarity search.
- Use the **Vector Store Retriever** node, regardless of the vector store used. We can:
  - We can specify how many documents to extract—they will be returned as a list.
  - Apply filters to narrow results based on metadata


2. Populate vector store
3. Retrieve rel
OpenAI Embedding Model Selector
FAISS Vector Store Creator
Vector Store Retriever
Document Data Extractor
Text Chunker
Text chunking
Add documents to vector store
Retrieve results
Access database data for analysis
SQLite Connector
Query DB data
alysis
the product and the ent into columns
Table Validator
Review analysis
Validate GenAI results before updating a table on Databricks

5. Retrieve relevant information from the vector store

Dialog - 4:1488 - Vector Store Retriever (Retrieve results)

Queries column: Review

Number of results: 3

Retrieved document column name: Retrieved documents

☑ Retrieve metadata from documents

Metadata Filter

Metadata Filter 1

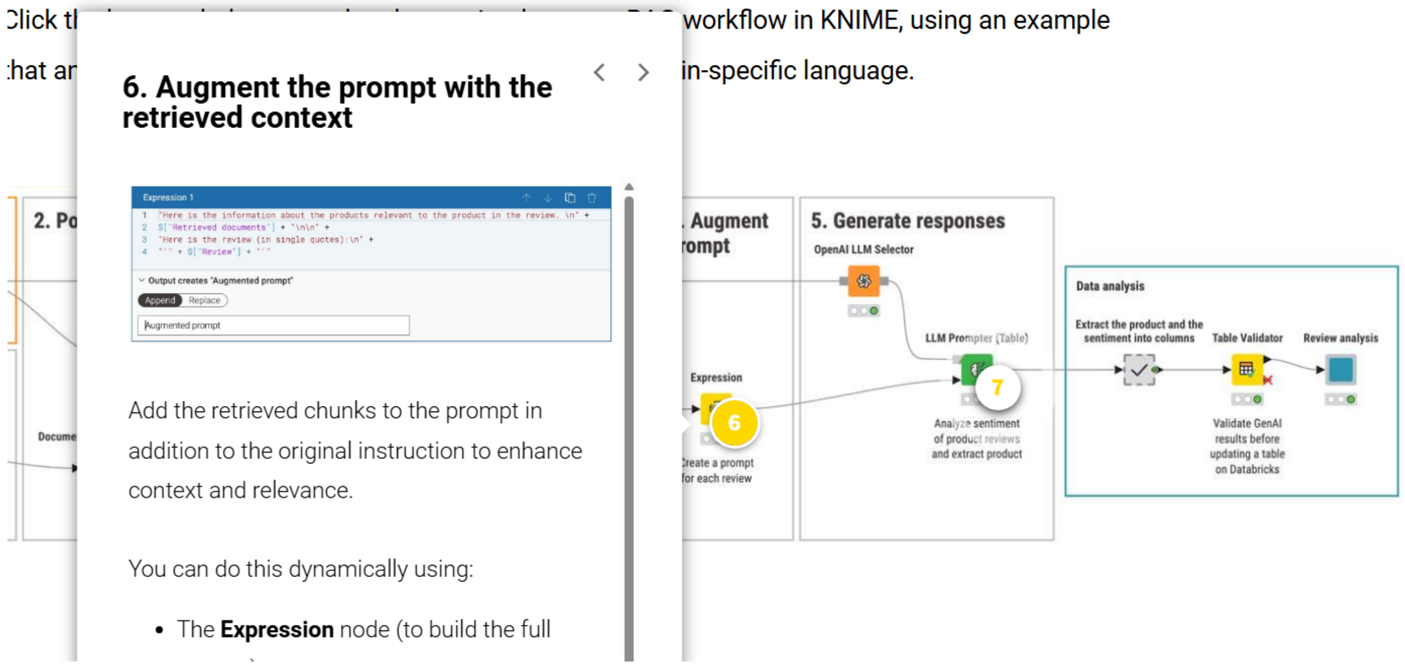
Metadata column: No values present

Metadata value:

Add filter

6- Augment the prompt

- Add the retrieved chunks to the prompt in addition to the original instruction to enhance context and relevance.
- We can do this dynamically using:
    - The **Expression** node (to build the full prompt),
    - The **Message Creator** node (to combine original prompt and retrieved chunks).



Click tl workflow in KNIME, using an example
hat ar in-specific language.

6. Augment the prompt with the retrieved context

Expression 1

```
"Here is the information about the products relevant to the product in the review. \n" +
$["Retrieved documents"] + "\n\n" +
"Here is the review (in single quotes):\n" +
"'" + $["Review"] + "'"
```

Output creates "Augmented prompt"

[✓ Append] Replace

Augmented prompt

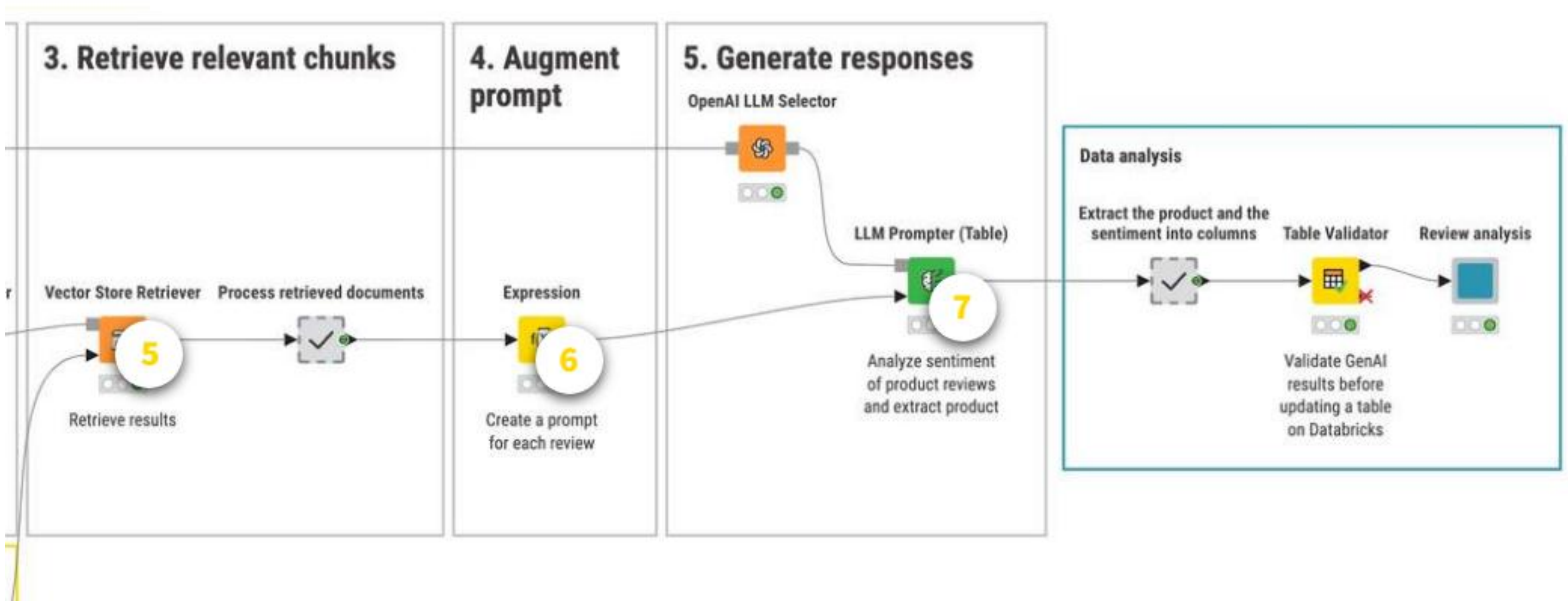
Add the retrieved chunks to the prompt in addition to the original instruction to enhance context and relevance.

You can do this dynamically using:

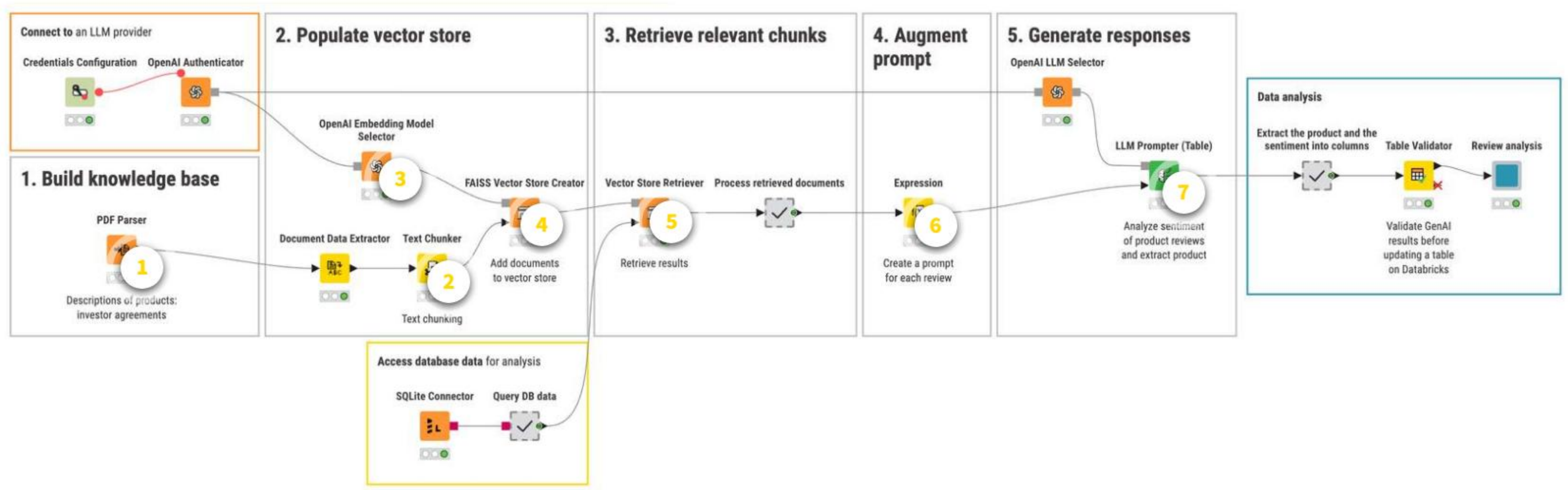
- The **Expression** node (to build the full

7- Submit the augmented prompt(s) to an LLM

- Send the augmented prompt as usual to the LLM and receive more tailored responses, enriched by the additional knowledge.


3. Retrieve relevant chunks
4. Augment prompt
5. Generate responses
OpenAI LLM Selector
Data analysis
Extract the product and the sentiment into columns
Table Validator
Review analysis
LLM Prompter (Table)
Vector Store Retriever
Process retrieved documents
Expression
5
6
7
Retrieve results
Create a prompt for each review
Analyze sentiment of product reviews and extract product
Validate GenAI results before updating a table on Databricks

RAG Example in KNIME



Connect to an LLM provider
Credentials Configuration
OpenAI Authenticator
1. Build knowledge base
PDF Parser
Descriptions of products: investor agreements
2. Populate vector store
OpenAI Embedding Model Selector
Document Data Extractor
Text Chunker
Text chunking
FAISS Vector Store Creator
Add documents to vector store
3. Retrieve relevant chunks
Vector Store Retriever
Process retrieved documents
Retrieve results
4. Augment prompt
Expression
Create a prompt for each review
5. Generate responses
OpenAI LLM Selector
LLM Prompter (Table)
Analyze sentiment of product reviews and extract product
Data analysis
Extract the product and the sentiment into columns
Table Validator
Review analysis
Validate GenAI results before updating a table on Databricks
Access database data for analysis
SQLite Connector
Query DB data
1
2
3
4
5
6
7

This workflow can be downloaded as following:

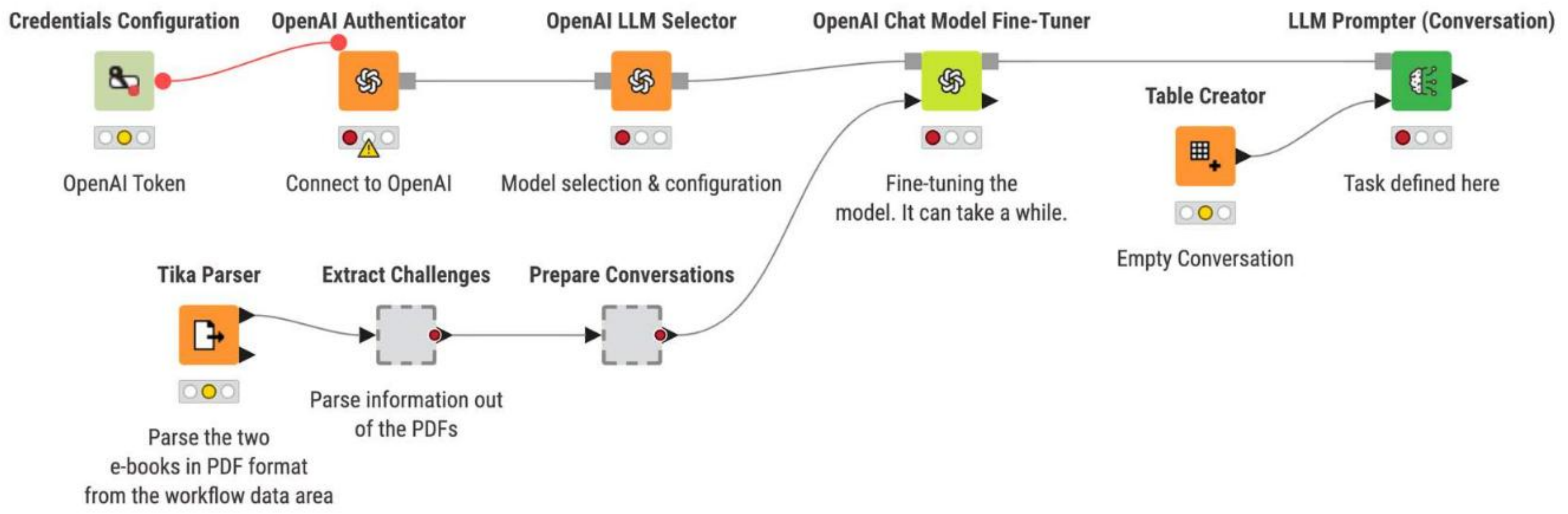
1. Download Course Workflows from VClass
2. Goto Generative AI Folder -> AI Extension Guide -> RAG
3. Open Product FAQ

Fine-tuning

- RAG has its limitations and can fail, for example, if the model lacks the foundational knowledge to simply understand the user’s question (with or without retrieved context).
- Another strategy we can use to reduce hallucinations in LLMs is to further train the model on new data that is related data to our task.
- Nevertheless, retraining an entire large language model from scratch is impractical because of their size.
- Another method we can use is called **Fine Tuning**.
- Fine Tuning is the process of training an existing language model on a smaller, task-specific dataset to improve performance on a particular domain or application.
- This adjusts the model's internal parameters based on the new data without retraining it from scratch.

Fine-Tuning

- When working with closed-source models (like OpenAI's), fine-tuning typically involves:
    1. Uploading your training data via an API call
    2. Letting the provider perform the fine-tuning on their infrastructure
    3. Accessing the newly fine-tuned model through a dedicated model ID via future API calls
- KNIME makes it possible to fine-tune OpenAI chat models through the dedicated node.
- You can also delete fine-tuned models from your OpenAI account when no longer needed.


Credentials Configuration
OpenAI Token
OpenAI Authenticator
Connect to OpenAI
OpenAI LLM Selector
Model selection & configuration
OpenAI Chat Model Fine-Tuner
Fine-tuning the model. It can take a while.
Table Creator
Empty Conversation
LLM Prompter (Conversation)
Task defined here
Tika Parser
Parse the two e-books in PDF format from the workflow data area
Extract Challenges
Parse information out of the PDFs
Prepare Conversations